Using WebSphere Application Server Optimized Local Adapters (WOLA) to Integrate COBOL and zAAP-able Java

David Follis
IBM

March 12, 2014
Session Number 14693
Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

- CICS*
- DB2*
- GDPS*
- Geographically Dispersed Parallel Sysplex
- HiperSockets
- IBM*
- IBM eServer
- IBM logo*
- IMS
- On Demand Business logo

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

- Java and all Java-related trademarks and logos are trademarks of Oracle.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
- SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.
- MIB is a trademark of MIB Group Inc.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.
Disclaimer

- The information contained in this documentation is provided for informational purposes only. While efforts were many to verify the completeness and accuracy of the information contained in this document, it is provided “as is” without warranty of any kind, express or implied.

- This information is based on IBM’s current product plans and strategy, which are subject to change without notice. IBM will not be responsible for any damages arising out of the use of, or otherwise related to, this documentation or any other documentation.

- Nothing contained in this documentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM (or its suppliers or licensors), or altering the terms and conditions of the applicable license agreement governing the use of the IBM software.

- Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

- All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.
<table>
<thead>
<tr>
<th>Session</th>
<th>Title</th>
<th>Time</th>
<th>Room</th>
<th>Speaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>14618</td>
<td>Getting Started with WebSphere Liberty Profile on z/OS</td>
<td>Monday 9:30</td>
<td>Grand Ballroom Salon C</td>
<td>Loos/Follis</td>
</tr>
<tr>
<td>14692</td>
<td>Getting Started with WebSphere Compute Grid</td>
<td>Tuesday 9:30</td>
<td>Grand Ballroom Salon J</td>
<td>Hutchinson/Loos</td>
</tr>
<tr>
<td>14693</td>
<td>Using WebSphere Application Server Optimized Local Adapters (WOLA) to Migrate Your COBOL to zAAP-able Java</td>
<td>Wednesday 9:30</td>
<td>Grand Ballroom Salon K</td>
<td>David Follis</td>
</tr>
<tr>
<td>14620</td>
<td>WebSphere Liberty Profile on Windows AND z/OS (among other things) Hands-on Lab</td>
<td>Wednesday 1:30</td>
<td>Platinum Ballroom Salon 7</td>
<td></td>
</tr>
<tr>
<td>14949</td>
<td>Tips Learned Implementing Websphere Application Server (WAS) on Linux for IBM System z</td>
<td>Wednesday 3:00</td>
<td>Grand Ballroom Salon G</td>
<td>Eberhard Pasch</td>
</tr>
<tr>
<td>14709</td>
<td>Need a Support Assistant? Check Out IBM's! (ISA)</td>
<td>Thursday 8:00</td>
<td>Grand Ballroom Salon A</td>
<td>Mike Stephen</td>
</tr>
<tr>
<td>15050</td>
<td>z/OSMF 2.1 Implementation and Configuration</td>
<td>Thursday 8:00</td>
<td>Grand Ballroom Salon G</td>
<td>Greg Daynes</td>
</tr>
<tr>
<td>14832</td>
<td>Web Apps using Liberty Profile Technology in CICS</td>
<td>Thursday 11:00</td>
<td>Platinum Ballroom Salon 2</td>
<td>Ian Mitchell</td>
</tr>
<tr>
<td>14722</td>
<td>Assimilating WebSphere Application Server into your z/OS WLM Configuration</td>
<td>Thursday 1:30</td>
<td>Orange County Salon 1</td>
<td>David Follis</td>
</tr>
<tr>
<td>15017</td>
<td>Using IBM WebSphere Application Server and IBM WebSphere MQ Together [z/OS &amp; Distributed]</td>
<td>Thursday 3:00</td>
<td>Grand Ballroom Salon A</td>
<td>Ralph Bateman</td>
</tr>
</tbody>
</table>
WebSphere Optimized Local Adapters

Introduction and Overview
In The Beginning: Cross Memory

WAS z/OS has had cross-memory from the beginning:

1 – Controller to Servant

A request queued to WLM is pulled from the CR to the SR using cross-memory services. What gets queued in WLM is really a token to the address location within the CR where the request is currently held.

2 – Server to Server

When a servlet in one server makes an IIOP call to an EJB in another server in the same cell and on the same LPAR, WAS z/OS recognizes this and bypasses the TCP stack. The IIOP call is made using cross-memory services.

This is called LOCALCOMM, and it's the key to starting the story of what WOLA is and how it works.

We're setting the stage for WOLA by providing historical context

WOLA is built on this from-the-beginning LOCALCOMM function
Original Requirement: Batch into WAS
Access Java asset in WAS from batch COBOL

Potential Solutions:

- **Use JNI and RMI/IIOP**
  Somewhat complex: require C/C++ stub for JNI; JNI code to interface to Java; JVM to host Java used to RMI/IIOP call into WAS z/OS; code to perform bootstrap and lookup, etc., etc.

- **Use MQ**
  Most obvious ... but would require target EJB to be MDB or know about MQ to PUT/GET; introduces an asynchronous nature that may not suit batch program.

- **Use Web Services**
  Challenging unless batch program has infrastructure to open sockets, understand HTTP, form SOAP or RESTful calls. Plus, web services is somewhat high-overhead with larger latency per call, which can harm batch processing window.

- **Something else?**

This is where WOLA came from
Externalized LOCALCOMM = WOLA

WOLA is programmatic interface to LOCALCOMM

The cross-memory function already existing within WAS z/OS
Key was to provide a way for external address spaces to access it

WOLA is just that – an infrastructure and interface for external address spaces to participate in the LOCALCOMM exchange mechanism of WAS z/OS

Not just COBOL batch ... that's just our starting point. Supports several different programming languages, CICS and IMS.
WOLA is Bi-Directional

This chart shows a few use-cases

1 – Inbound from Batch
This is the use-case we said sparked the development of WOLA. We see this in cases where batch programs need data from EJB assets, particularly ISV apps.

2 – Inbound from CICS or IMS
This is in many ways the same as inbound from batch, though with CICS or IMS you have the added benefit of transaction and security propagation.

3 – Outbound to CICS
The key benefit of this is it allows channels/containers over a cross-memory exchange. CTG EXCI is limited to 32K COMMAREA, and CTG channels/containers provided over IPIC.

Note: CTG has other benefits related to HA designs and cross-LPAR communications. We are not saying WOLA replaces CTG; it complements it.

4 – Outbound to Batch
This is an interesting use-case we’re seeing more of lately… it provides a means of exposing existing batch programs as a service, using web services or any other access method supported by WAS itself.
Value of WOLA

WOLA has several strong technical value attributes

- Low Latency
  - Bypasses TCP stack; LOCALCOMM code path relatively lightweight; lowest per-call latency available

- Large Messages
  - Provides efficient means of passing large messages (up to 2GB) including CICS channels/containers and IMS multi-segment/large-message support; avoids overhead of segmenting and rebuilding messages.

- Secure Exchange
  - Cross-memory never touches network stack or network infrastructure so it can’t be sniffed and it can’t be altered in flight.

- Transaction Propagation
  - Ability to propagate TX with CICS or IMS so global transactions using z/OS RRS as synchpoint coordinator

- Security Propagation
  - Ability to propagate identity with CICS or IMS without requiring coding of aliases
Integrating Java / Non-Java Using WOLA

Enabling some of your GP work to be refactored to Java

WOLA is a technology that enables integration of Java and non-Java assets in a way that provides high throughput and security.
WebSphere Optimized Local Adapters

Outbound WOLA to CICS
Accessing CICS from WAS z/OS

Many different ways this can be done

Some considerations:

- CTG is a separately licensed product
- EXCI limited to 32K COMMAREA
- CTG IPIC allows >32K and involves TCP/IP
- MQ is asynchronous
- Web Services may have higher overhead

WOLA:

- Comes with WAS z/OS
- COMMAREA or Chan/Containers
- TX and Security Propagation
- Cross Memory latency
WOLA Support and CICS

Here’s what outbound WOLA to CICS looks like

WOLA supplies a JCA resource adapter, which application uses to access WOLA services.

WOLA “registration” is the cross-memory connection over which communications flow.

WOLA TRUE and Link Server provide support in CICS region and hides target CICS from any knowledge of WOLA.
Customer Use-Case Example

**WOLA as part of complex stock trading architecture**

<table>
<thead>
<tr>
<th>Trade Processing System - Context</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mainframe - Z/OS</strong></td>
</tr>
<tr>
<td>CICS</td>
</tr>
<tr>
<td>Trade Figuration System</td>
</tr>
<tr>
<td>OLA</td>
</tr>
<tr>
<td>VSAM File (Trade Activity)</td>
</tr>
<tr>
<td>TPS Request Queue</td>
</tr>
<tr>
<td>Pre-Standard Message</td>
</tr>
<tr>
<td>TPS ProcessBean</td>
</tr>
<tr>
<td><strong>Websphere Application Server</strong></td>
</tr>
<tr>
<td>JMS/JCA</td>
</tr>
<tr>
<td>JMS: Request MDB</td>
</tr>
<tr>
<td>TPSDataBeans</td>
</tr>
<tr>
<td>- PROMDataBean</td>
</tr>
<tr>
<td>- AccountDataBean</td>
</tr>
<tr>
<td>- TRAMUpdateBean</td>
</tr>
<tr>
<td>TPSVSAMBean</td>
</tr>
<tr>
<td><strong>TPSCICSBean</strong></td>
</tr>
<tr>
<td><strong>DB2</strong></td>
</tr>
<tr>
<td>JCA Connector</td>
</tr>
<tr>
<td>JDBC</td>
</tr>
<tr>
<td>JRIO</td>
</tr>
</tbody>
</table>

Proof of Concept evaluated WOLA compared to CTG

WOLA shows better performance for COMMAREA and Channels/Containers

Eliminates need for CTG for this part of the architecture

WOLA used as one part of larger-scale application topology
Multiple CICS Regions (Gateways)

WOLA supports multiple concurrent registrations

WOLA to multiple CICS regions serving as gateway regions to larger CICSplex

Round-robin support added in 8.0.0.1

Provides highly utilized WOLA connections with balancing and HA
Network clients access business functions through WAS z/OS, which turns and drives CICS function over WOLA implemented with round-robin support to CICS gateway regions.
Development Mode Support

Allows Java developers greater dev/test flexibility

Developer is free to deploy, test, change settings on development workstation WAS

To application WOLA is in use, but in reality call is mapped to network

WOLA Proxy EJB intercepts and makes “real” WOLA call to CICS
WOLA and IBM RAD 9.0.1

RAD 9.0.1 includes integrated tooling support for WOLA

Note: This new RAD 9.0.1 WOLA support applies to WAS 8.5.0.+ levels, which is the level embedded with V9 RAD. The RAD support can be used on prior levels of WAS z/OS, but users will need to bring in the associated level's olaapis.jar and ola.rar resource adapter and place it on their RAD build path.

IBM Webpage with details of RAD 9.0.1 Features and Functions

IBM developerWorks article showing end-to-end use case scenario
WOLA and CICS Summary Chart

One chart showing supported functionality

The registration into WAS must have the appropriate TXN and SEC settings to support propagation of global transaction and propagation of security identity.

Complete your session evaluations online at www.SHARE.org/AnaheimEval
WebSphere Optimized Local Adapters

WOLA and IMS
WOLA Works with IMS

Here’s a quick summary chart
Summary of WOLA Capabilities and IMS

Particularly TX and Security Assertion
WebSphere Optimized Local Adapters

Inbound WOLA and APIs
Key Elements of the Inbound Design

Target EJB and use of the WOLA APIs

COBOL, C/C++, PL/I or High Level Assembler

WOLUModules copied out from file system using copyZOS.sh

//STEPLIB DD DSN=

Control Regions

• Stateless EJB
• Implements execute()
• Interfaces implemented with supplied WOLA classes

The program outside of WAS uses the WOLA APIs to register and invoke the target EJB and get a response. The target EJB has a few easy-to-implement requirements. That EJB may call other EJBs as part of its processing.
This article spells out the details of each of the 13 WOLA APIs. However, it implies all 13 need to be used, which is not true. Over the next two charts we’ll see how to organize the APIs into categories of usage.
**APIs ... Basic vs. Advanced**

*Basic APIs are simple to use but make assumptions*

---

**BBOA1INV**
*Invoke EJB*

- One API is all you need to invoke an EJB in WAS z/OS
- BBOA1INV makes assumptions to keep things simple
  - Program control held until response received
  - Maximum response length predictable

**BBOA1CNG**
*Get connection*

**BBOA1SRQ**
*Send request*

**BBOA1GET**
*Get response*

**BBOA1CNR**
*Release connection*

**BBOA1RCL**
*Response length*

---

**Advanced APIs do the same function as basic, but with more control given to you**

---

If the “basic” APIs suit your needs, then use them. But if you need more control, then use “advanced.” At the end of the day the same result is achieved – use of WOLA to communicate between address spaces.
How the APIs Organize

Inbound, outbound ... basic and advanced
Tax Calculation on Web Service Platform

Real example of WOLA usage with advanced APIs

Advanced inbound APIs with asynchronous control
Asynchronous because COBOL is single-threaded and web service call to external tax package is the slowest link. Asynchronous APIs allows COBOL to get program control immediately.

150 connections kept loaded with work and busy
Maximum connections over WOLA to EJB. All 150 loaded up with work requests. COBOL then loops through array to see if response received. If so, then process back results and load that connection with another request. Connections kept fully busy in this manner.

Multi-threaded Java then parallelized web service calls
WAS z/OS and WAS distributed are multi-threaded. Given sufficient processing capacity, the work requests from COBOL may then be handled in a parallel execution fashion.

Complete your session evaluations online at www.SHARE.org/AnaheimEval
WebSphere Optimized Local Adapters

Summary
Summary of WOLA Usage and Value

A few points to consider

WOLA is bi-directional, which means you can use it to build solutions that go outbound from WAS z/OS, or come inbound to WAS z/OS

With CICS and IMS WOLA supports propagation of transaction and security, giving you flexibility to construct solution to your needs

Because WOLA is cross-memory, it provides an extremely low per-call latency profile. In high-volume, repetitive environments latency adds up.

WOLA provides a mechanism to better integrate Java and non-Java, which may allow you to move work to Java and reduce GP usage:

• Move business logic from CICS to WAS z/OS and use WOLA
• Move business logic from batch to WAS z/OS and use WOLA
• Take advantage of ISV solutions in WAS z/OS and integrate with WOLA
WebSphere z/OS Optimized Local Adapters

Abstract: WebSphere Optimized Local Adapters (or "WOLA") is a high-speed memory-to-memory transfer technology function provided with WebSphere Application Server for z/OS. It provides an excellent mechanism for communicating between WAS z/OS and other systems such as CICS, IMS and Batch programs. WOLA first came available in 7.0.0.4 and has been enhanced several times since then.

Executive Introduction and Overview
The following two-page brochure provides a good overview of WOLA. When printed duplex in color on good paper stock, it makes an excellent handout for meetings and events.

Overview material
History of functional updates
API coding Primer
WOLA Videos